IMPERIAL COLLEGE LONDON

Stochastic Control in Finance Project: Merton Portfolio Optimisation with Consumption

Tianhao Wang, Ben Somner-Bogard

September 4, 2022

The software and packages used for this project including *Python*[1] with *tensorflow* [2], *sklearn*[3] and *pandas*[4] packages.

The reference used for this project is our lecture slides and handout.

1 Introduction

Define the valued control process $\mathbf{u} \in \mathbb{R} \times \mathbb{R}_+$ as

$$\mathbf{u} = \left([\pi_t, c_t] \right)_{0 \le t \ge T}. \tag{1.1}$$

We start with $X^{\mathbf{u}}$ a controlled Markov diffusion, for $\mathbf{u}$ as defined in 1.1:

$$dX_t^{\mathbf{u}} = \mu(t, X_t^{\mathbf{u}}, \mathbf{u}_t)dt + \sigma(t, X_t^{\mathbf{u}}, \mathbf{u}_t)dB_t \tag{1.2}$$

where

$$\mu(t, x, \mathbf{u}) = [r + (\mu - r)\pi_t] \cdot x - c, \quad \sigma(t, x, \mathbf{u}) = \sigma \pi_t x. \tag{1.3}$$

with $\mathbf{u} = (\pi, c) \in \mathbb{R} \times \mathbb{R}_+$. We aim to maximise the following performance criterion:

$$H^{\mathbf{u}}(t, x) = \sup_{\mathbf{u} \in \mathcal{A}_{t,T}} \mathbb{E}\left[\int_0^T e^{-\rho t} u_1(c_t) dt + e^{-\rho T} u_2(X_T^{\mathbf{u}}) \right], \tag{1.4}$$

where

$$u_2(x) = K \cdot \frac{x^{1-\gamma}}{1-\gamma}, \quad u_1(c) = \frac{c^{1-\gamma}}{1-\gamma}, \tag{1.5}$$

with $\mathbf{u} = (\pi, c) \in \mathbb{R} \times \mathbb{R}^+$. For the full problem, see the project specification.

2 Solving the Control Problem

We follow the five step "algorithm" described in lectures.

2.1 Step 1: Write down the Hamilton-Jacobi-Bellman Equation

We have seen a form of the HJB equation in lectures but in order to apply it to the problem we have, we must first derive a more general version.

One can easily see that the HJB equation for this more general for will be very similar to the HJB equation seen in lectures (Week 3) with a correction term, to account for the consumption. In particular, the HJB equation reads as follows:

$$\sup_{\mathbf{u}\in\mathscr{A}_{t,T}}\left\{\partial_t H+\mu(t,x,\mathbf{u})\cdot\partial_x H+\frac{1}{2}\sigma^2(t,x,\mathbf{u})\cdot\partial_{xx}H+\rho\cdot H+u_1(c_t)\right\}=\rho\cdot H(t,x) \tag{2.1}$$

$$H(T,x)=u_2(x). \tag{2.2}$$

We now formalise the derivation of this more general formula.

Firstly, one can rewrite $H^{\mathbf{u}}(t,x)$, for $0\le t<t^*<T$, as:

$$H^{\mathbf{u}}(t,X_t^{\mathbf{u}})=\sup_{\mathbf{u}\in\mathscr{A}_{t,T}}\mathbb{E}\left[\int_t^T e^{-\rho(s-t)}u_1(c_s)ds+e^{-\rho(T-t)}u_2(X_T^{\mathbf{u}})|\mathscr{F}_t\right] \tag{2.3}$$

$$=\sup_{\mathbf{u}\in\mathscr{A}_{t,T}}\mathbb{E}\left[\int_t^{t^*} e^{-\rho(s-t)}u_1(c_s)ds \tag{2.4}$$

$$+\int_{t^*}^T e^{-\rho(s-t)}u_1(c_s)ds+e^{-\rho(T-t)}u_2(X_T^{\mathbf{u}}))|\mathscr{F}_t\right]$$

$$=\sup_{\mathbf{u}\in\mathscr{A}_{t,T}}\mathbb{E}\left[\int_t^{t^*} e^{-\rho(s-t)}u_1(c_s)ds \tag{2.5}$$

$$+e^{t^*-t}\left(\int_{t^*}^T e^{-\rho(s-t^*)}u_1(c_s)ds+e^{-\rho(T-t^*)}u_2(X_T^{\mathbf{u}}))\right)|\mathscr{F}_t\right]$$

$$=\sup_{\mathbf{u}\in\mathscr{A}_{t,T}}\mathbb{E}\left[\int_t^{t^*} e^{-\rho(s-t)}u_1(c_s)ds+e^{-\rho(t^*-t)}H^{\mathbf{u}}(t^*,X_{t^*}^{\mathbf{u}})|\mathscr{F}_t\right] \tag{2.6}$$

$$=e^{\rho t}\sup_{\mathbf{u}\in\mathscr{A}_{t,T}}\mathbb{E}\left[\int_t^{t^*} e^{-\rho s}u_1(c_s)ds+e^{-\rho t^*}H^{\mathbf{u}}(t^*,X_{t^*}^{\mathbf{u}})|\mathscr{F}_t\right] \tag{2.7}$$

Then the HJB condition that we want to solve, in differential form, is:

$$0=\sup_{\mathbf{u}\in\mathscr{A}_{t,T}}\mathbb{E}\left[e^{-\rho t}u_1(c_t)dt+d(e^{-\rho t}H^{\mathbf{u}}(t,X_t^{\mathbf{u}}))|\mathscr{F}_t\right] \tag{2.8}$$

$$\rho H^{\mathbf{u}}(t,X_t^{\mathbf{u}})dt=\sup_{\mathbf{u}\in\mathscr{A}_{t,T}}\mathbb{E}\left[u_1(c_t)dt+d(H^{\mathbf{u}}(t,X_t^{\mathbf{u}}))|\mathscr{F}_t\right] \tag{2.9}$$

Since $X_t^{\mathbf{u}}$ is an Itô diffusion (defines in 1.2 and 1.3), we can apply Itô$'s$ formula to $H^{\mathbf{u}}(t, X_t^{\mathbf{u}})$. I.e.

$$dH(t, X_t^{\mathbf{u}}) = \left(\partial_t H + \mu_t \partial_x H + \frac{1}{2}\sigma_t^2 \partial_{xx} H\right) dt + \sigma_t \partial_x H dB_t \tag{2.10}$$

Which gives:

$$\rho H^{\mathbf{u}}(t, X_t^{\mathbf{u}}) dt = \sup_{\mathbf{u} \in \mathscr{A}_{t,T}} \left((\partial_t H + \mu_t \partial_x H + \frac{1}{2}\sigma_t^2 \partial_{xx} H) dt + u_1(c_t) dt \right) \tag{2.11}$$

$$\rho H^{\mathbf{u}}(t, X_t^{\mathbf{u}}) = \sup_{\mathbf{u} \in \mathscr{A}_{t,T}} \left((\partial_t H + \mu_t \partial_x H + \frac{1}{2}\sigma_t^2 \partial_{xx} H) + u_1(c_t) \right) \tag{2.12}$$

Finally, substituting our formulae for $\mu_t, \sigma_t and u_1$, we obtain:

$$\sup_{\mathbf{u} \in \mathscr{A}_{t,T}} \left\{ \partial_t H + ([r + (\mu - r)\pi_t] \cdot x - c_t)\partial_x H + \frac{1}{2}\sigma^2 \pi_t^2 x^2 \partial_{xx} H + \frac{c_t^{1-\gamma}}{1-\gamma} \right\} = \rho H^{\mathbf{u}}(t, X_t^{\mathbf{u}}) \tag{2.13}$$

$$H(T, x) = K \cdot \frac{x^{1-\gamma}}{1-\gamma}. \tag{2.14}$$

We have now completed step 1.

2.2 Step 2: Find maximiser $\mathbf{u} = (\pi^*, c^*)$ in HJB equation

We first differentiate 2.13 with respect to π, treating everything else as constant.

$$(\mu - r) \cdot x \cdot \partial_x H + \sigma^2 \pi x^2 \partial_{xx} H) = 0 \tag{2.15}$$

$$(\mu - r) \cdot \partial_x H + \sigma^2 \pi x \partial_{xx} H) = 0 \tag{2.16}$$

$$\implies \quad \pi^*(t, x) = \frac{\partial_x H(r - \mu)}{\sigma^2 x \partial_{xx} H} \tag{2.17}$$

Now we differentiate 2.13 with respect to c, treating everything else as constant.

$$-\partial_x H + (1-\gamma)\frac{c^{-\gamma}}{1-\gamma} = 0 \tag{2.18}$$

$$-\partial_x H + c^{-\gamma} = 0 \tag{2.19}$$

$$\implies \quad c^*(t, x) = (\partial_x H)^{-\frac{1}{\gamma}} \tag{2.20}$$

2.3 Step 3: Substitute π^* and c^* back into our HJB equation

We substitute our formula for π_t^* (2.17) and out formula for c_t^* (2.20) into our HJB equation (2.13):

$$\partial_t H + ([r + (\mu - r)\frac{\partial_x H(r-\mu)}{\sigma^2 x \partial_{xx} H}] \cdot x - (\partial_x H)^{-\frac{1}{\gamma}})\partial_x H + \frac{1}{2}\sigma^2 \frac{\partial_x H(r-\mu)}{\sigma^2 x \partial_{xx} H}^2 x^2 \partial_{xx} H + \frac{(\partial_x H)^{-\frac{1}{\gamma}1-\gamma}}{1-\gamma} = \rho H \tag{2.21}$$

$$\partial_t H + r x \partial_x H - \frac{(\mu-r)^2}{2\sigma^2}\frac{(\partial_x H)^2}{\partial_{xx} H} + \frac{\gamma}{1-\gamma}(\partial_x H)^{1-\frac{1}{\gamma}} = \rho H \tag{2.22}$$

$$H(T,x) = K \cdot \frac{x^{1-\gamma}}{1-\gamma}. \tag{2.23}$$

In order for our optimisation to produce a maximum we need that $\gamma > 0$ (to ensure that the consumption function is concave) $\partial_{xx} H < 0, X_t^{\mathbf{u}} > 0 and c_t^* > 0 \forall 0 \le t < T$.

2.4 Step 4: Solve HJB equation for H

As hinted in lectures, we make the same guess as with the standard Merton problem. Namely

$$H(t,x) = f(t)g(x) \tag{2.24}$$

$$s.t. H(T,x) = f(T)g(x) = K \cdot \frac{x^{1-\gamma}}{1-\gamma} \tag{2.25}$$

$$\implies f(T) = K, \quad g(x) = \frac{x^{1-\gamma}}{1-\gamma}. \tag{2.26}$$

Remark (Derivatives of H(t,x) guess)**.**

$$\partial_t H = f'(t) \cdot \frac{x^{1-\gamma}}{1-\gamma} \tag{2.27}$$

$$\partial_x H = f(t) \cdot x^{-\gamma} \tag{2.28}$$

$$\partial_{xx} H = f(t) \cdot (-\gamma) x^{-\gamma-1}. \tag{2.29}$$

Substituting our guess and the associated derivatives into our PDE gives:

$$f'(t) \cdot \frac{x^{1-\gamma}}{1-\gamma} + r f(t) \cdot x^{1-\gamma} + \frac{(\mu-r)^2}{2\gamma\sigma^2} f(t) \cdot x^{1-\gamma} + \frac{\gamma}{1-\gamma}(f(t)^{1-\frac{1}{\gamma}} \cdot x^{1-\gamma}) = \rho f(t) \cdot \frac{x^{1-\gamma}}{1-\gamma} \tag{2.30}$$

$$\implies f'(t) + \frac{-2\rho\gamma\sigma^2 + (1-\gamma)[(\mu-r)^2 + 2r\gamma\sigma^2]}{2\gamma\sigma^2} f(t) + \gamma f(t)^{1-\frac{1}{\gamma}} = 0 \tag{2.31}$$

with boundary condition $f(T) = K$.
We assume that $f(t) = h(t)^\beta$ with h a function of time and β a constant to be determined.

$$\implies f'(t) = \beta \cdot h(t)^{\beta-1} \cdot h'(t) \tag{2.32}$$

Substituting this into equation 2.31 yields:

$$\beta \cdot h(t)^{\beta-1} \cdot h'(t) + \frac{-2\rho\gamma\sigma^2 + (1-\gamma)[(\mu-r)^2 + 2r\gamma\sigma^2]}{2\gamma\sigma^2} h(t)^\beta + \gamma h(t)^{\beta(1-\frac{1}{\gamma})} = 0 \tag{2.33}$$

We choose to set $\beta = \gamma$:

$$\gamma \cdot h(t)^{\gamma-1} \cdot (h'(t) + 1) + \frac{-2\rho\gamma\sigma^2 + (1-\gamma)[(\mu-r)^2 + 2r\gamma\sigma^2]}{2\gamma\sigma^2} h(t)^\gamma = 0 \tag{2.34}$$

Letting $C := \frac{\rho - (1-\gamma)\left[\frac{(\mu-r)^2}{2\gamma\sigma^2} + r\right]}{\gamma}$, we obtain the following ODE:

$$h'(t) = C \cdot h(t) - 1 \tag{2.35}$$

with boundary condition $h(T) = K^{\frac{1}{\gamma}}$. This ODE is easily solved. We have two cases:

Case 1: $C = 0$, then $h'(t) = -1$ with $h(T) = K^{\frac{1}{\gamma}}$. This has trivial solution $h(t) = K^{\frac{1}{\gamma}} + T - t$.

Case 2: $C \neq 0$, then $h'(t) = C \cdot h(t) - 1$ with $h(T) = K^{\frac{1}{\gamma}}$. This is a separable ODE and can be solved as follows:

$$\frac{h'(t)}{C \cdot h(t) - 1} = 1 \tag{2.36}$$

$$\int_t^T \frac{h'(t)}{C \cdot h(t) - 1} dt = \int_t^T 1 dt \tag{2.37}$$

$$\frac{\ln(C \cdot h(t) - 1)}{C} = t + b \tag{2.38}$$

$$h(t) = \frac{e^{C(t+b)} + 1}{C} \tag{2.39}$$

Using the boundary condition, we obtain $b = \frac{\ln(CK^{\frac{1}{\gamma}} - 1)}{C} - T$.
Putting these together, we get h as follows:

$$h(t) = \begin{cases} K^{\frac{1}{\gamma}} + T - t & C = 0 \\ \frac{(CK^{\frac{1}{\gamma}} - 1)e^{c(t-T)} + 1}{C} & o.w. \end{cases} \tag{2.40}$$

This completes step 4.

2.5 Step 5: Re-evaluate π^* and c^* to eliminate H

We now substitute this formula for H back into our formulae for π^* (2.17) and c^* (2.20).

Remark (Derivatives of H(t,x))**.** *We have*

$$\partial_t H = \gamma \cdot h(t)^{\gamma-1} \cdot (C \cdot h(t) - 1) \cdot \frac{x^{1-\gamma}}{1-\gamma} \tag{2.41}$$

$$\partial_x H = h(t)^{\gamma} \cdot x^{-\gamma} \tag{2.42}$$

$$\partial_{xx} H = h(t)^{\gamma} \cdot (-\gamma) x^{-\gamma-1}. \tag{2.43}$$

Starting with π^*:

$$\pi^*(t,x) = \frac{\partial_x H(r-\mu)}{\sigma^2 x \partial_{xx} H} \tag{2.44}$$

$$= \frac{\cancel{h(t)^{\gamma}} \cdot \cancel{x^{-\gamma}}(r-\mu)}{\sigma^2 \cancel{x} \cancel{h(t)^{\gamma}} \cdot (-\gamma) \cancel{x^{-\gamma-1}}} \tag{2.45}$$

$$\boxed{\pi^*(t,x) = \frac{\mu - r}{\sigma^2 \gamma}} \tag{2.46}$$

Now c^*:

$$c^*(t,x) = (\partial_x H)^{-\frac{1}{\gamma}} \tag{2.47}$$

$$= (h(t)^{\gamma} \cdot x^{-\gamma})^{-\frac{1}{\gamma}} \tag{2.48}$$

$$= h(t)^{-1} \cdot x \tag{2.49}$$

$$\boxed{c^*(t,x) = \begin{cases} \frac{x}{K^{\frac{1}{\gamma}} + T - t} & C = 0 \\ \frac{Cx}{(CK^{\frac{1}{\gamma}} - 1)e^{C(t-T)} + 1} & o.w. \end{cases}} \tag{2.50}$$

where recall that constant $C \in \mathbb{R}$ represents

$$C = \frac{\rho - (1-\gamma)\left[\frac{(\mu-r)^2}{2\gamma\sigma^2} + r\right]}{\gamma} \tag{2.51}$$

Putting this all together gives the following optimal value function

$$H^*(t,x) = f(t)g(x) = h(t)^\gamma g(x) = \begin{cases} (K^{\frac{1}{\gamma}} + T - t)^\gamma \frac{x^{1-\gamma}}{1-\gamma} & C = 0 \\ \left(\frac{(CK^{\frac{1}{\gamma}} - 1)e^{C(t-T)} + 1}{C}\right)^\gamma \frac{x^{1-\gamma}}{1-\gamma} & o.w. \end{cases} \tag{2.52}$$

One should note that, to ensure that the constraints $X_t > 0$ and $c^*(t,x) > 0$, as well as the second order constraint on the second derivative, with respect to x, of H we need $h(t) > 0, \forall\, t \in [0,T)$. In particular, this ensures that $\frac{\partial^2 H^*}{\partial x^2} < 0$ so that we are indeed maximising the dynamic value function. One should note, that from the derivation and final formulae we need $0 < \gamma$ and $\gamma \neq 1$.

3 Qualitative Optimal Solution Behaviour

We look at the qualitative behaviour of some relevant quantities.

3.1 Qualitative behaviour of the controls

3.1.1 π^*: Optimal Allocation

By 2.46, the optimal proportion of wealth invested in the risky asset is neither dependent on time, t, nor the portfolio wealth, X_t. Intuitively, this makes sense, the higher the excess return, $(\mu - r)$, is, the more profitable the risky asset is, hence the optimal agent tends to invest more in the risky asset; whereas the higher the volatility, σ, is, the riskier the underlying asset is, so an agent acting optimally would invest less. Similarly, as γ increases, which measures the risk aversion of the agent, the allocation towards the risky asset will decrease.

As the underlying asset S_t follows a simple geometric Brownian motion with μ and σ, the agent does not have any extra information at each time t in the market, as the expected future value is the same as today's value due to the process being a martingale. Thus, it does not counter the agents decision to hold a constant proportion π^* over time, given different account balance X_t. The problem the agent faces and the information the agent possesses toward the market at each time t is the same as they face at time 0, with their current balance X_t being the only difference. Therefore, the proportion of their wealth invested into the risky asset, π^*, should stay constant at any time t, if the agent is acting optimally at each time t.

One interesting thing to note here is that, recalling the mean-variance goal function in portfolio management,

$$J = \mathbb{E}[X_T] - \frac{\gamma}{2}\mathbb{V}[X_T]$$

where γ, again, represents the risk aversion of the agent, the optimal investment into the risky asset is the same as 2.46. Note that consumption has not been considered here. One hypothesis is that the agent in the Merton problem with consumption might actually split their total wealth into pure investment (risky asset only) and pure bank account (bond and consumption) parts. When making the decision of what proportion of their total wealth to invest in the risky asset, the risk-free bond and consumption, the components are viewed as a group, hence the decision on consumption will not influence the proportion invested in the risky asset.

This hypothesis also hints to the reason why the hyper-parameter K is not considered when deciding the proportion of wealth to invest in the risky asset π^*. This analysis can be found in the next subsection. But in short, it is because K measures how much the agent cares about the terminal wealth X_T, instead of the terminal wealth from the risky asset $\pi^* \cdot X_T$.

3.1.2 c^*: Optimal Consumption

2.50 shows that c^* is positively related to the current balance X_t. Intuitively, the agent consumes more when they are richer, and at any time t, the agent is consuming a certain proportion of their current balance X_t, so that they can still leave some money in the market to provide some funds for their future consumption.

The exact proportion of X_t positively depends on the weight of future utility ρ and K. If the agent is impatient (larger ρ) and/or cares less (smaller K) about the terminal balance X_T, they tend to consume more over time. This effect tends to be weaker with larger γ and/or the market has lower sharp ratio $\frac{(\mu-r)}{\sigma}$, in which the agent is more reluctant to take risks and/or the risky asset is less valuable to invest in. In both cases, the agent would tend to invest less into the risky asset. As we analysed in the previous subsection, the optimal agent tends to view their proportions of wealth invested in pure investment (risky asset only) and in the bank account (bond and consumption) separately. As they invested less, the agent has more cash in the bank to consume, given no change in the risk-free bound return r.

The dependence of c^* on time t, however, is related to the other hyper-parameters. If the constant $C < 0$ (2.51), meaning that the importance of the investment to the agent (sharp ratio plus r WITH the agent's risk aversion being considered) overcomes the agents impatience (ρ) over time, c^* will be positively related to the current time t, i.e., the agent tends to delay their consumption so that they reserve more more money for investment.

On the other hand, when $C \geq 0$, the agent will start to hesitate between consumption and investment as the market is not very important or the agent is being too impatient. This effect will be amplified by hyper-parameter K. I.e., Once $C > 0$, meaning that the impatience of the agent overcomes the importance of the investment, the agent tends to seize the opportunity by consuming more at the beginning if K is small (in particular, $K < C^\gamma$), since the agent does not care that much about the terminal wealth X_T. And vice versa.

One thing needs to note is that C^* by construction is a fraction of X_t at any time t. This makes sure that $X_t > 0$ as long as $X_0 > 0$. This property ensures that some technical details in the previous sections, for example, utility functions u_1 and u_2 do not return complex number, so that ensures the optimal strategy is always admissible, as required under the scope of stochastic control problem.

3.2 Qualitative behaviour of the optimal value function

By 2.52, we know the optimal value function H^* is a function of time t and current wealth X_t. Further, notice the similarity (a v.s. $\frac{1}{a}$) of the coefficient part of the value function H^* and that of c^* (2.50), one can quickly conclude a similar analysis as we had in section 3.1.2 with a inverse effect for each hyper-parameter. For instance, the value function increases as the sharp ratio $\frac{\mu-r}{\sigma}$ decreases, hinting that if the risky asset becomes less salient, the same state (t, X_t) would become less attractive and hence the agent would tend to consume more.

In particular, the optimal value function $H^*(t, X_t)$ is a fraction of $u_2(X_t)$ but nothing to do with the $u_1(c)$. This does make sense once we think about it. The optimal c^* 2.20 is eventually a fraction of X_t. Hence, focusing on the X_t as well as the coefficient part of c^* before X_t is enough to conclude the value of current state (t, X_t) for decision.

4 Simulations

In the following we use Monte Carlo simulations in order to study the behaviour and optimality of our solution.

4.1 Simulation framework

We have programmed a simple Monte Carlo routine in python in order to simulate the dynamics and strategies of the problem. The details for the reasoning behind why we chose these hyper-parameters and the setting of the environment can be found in section 5

4.2 Simulation of Markov controls that differ from our optimal solution

We simulate the use of the following controls and study the expected final value of the payoff. If the supposed optimal strategy is indeed optimal it should have a higher expected final value when compared to any other strategy. Here, we have used the following parameter values:

Parameter	Value
μ	0.06
σ	0.4
T	10
x	1
r	0.03
ρ	0.01
γ	0.5
K	0.2
# of timesteps	1000
# of paths	1000

Table 4.1. hyper-parameters

4.2.1 Optimal Strategy

Recall, from 2.5, our optimal solution was

$$\pi^*(t,x) = \frac{\mu - r}{\sigma^2 \gamma} \tag{4.1}$$

$$c^*(t,x) = \begin{cases} \frac{x}{K^{\frac{1}{\gamma}} + T - t} & c = 0 \\ \frac{cx}{(cK^{\frac{1}{\gamma}} - 1)e^{c(t-T)} + 1} & o.w. \end{cases} \tag{4.2}$$

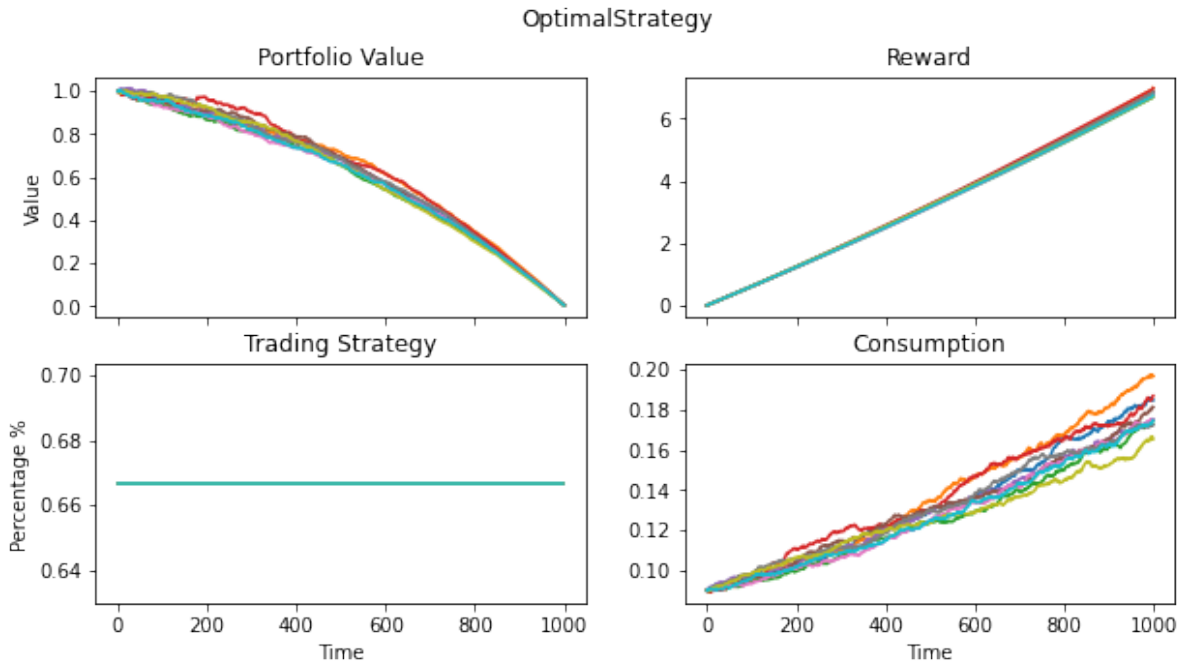


Figure 4.1. Evolution of the portfolio value and the reward for our optimal strategy through time, for 10 sampled paths.

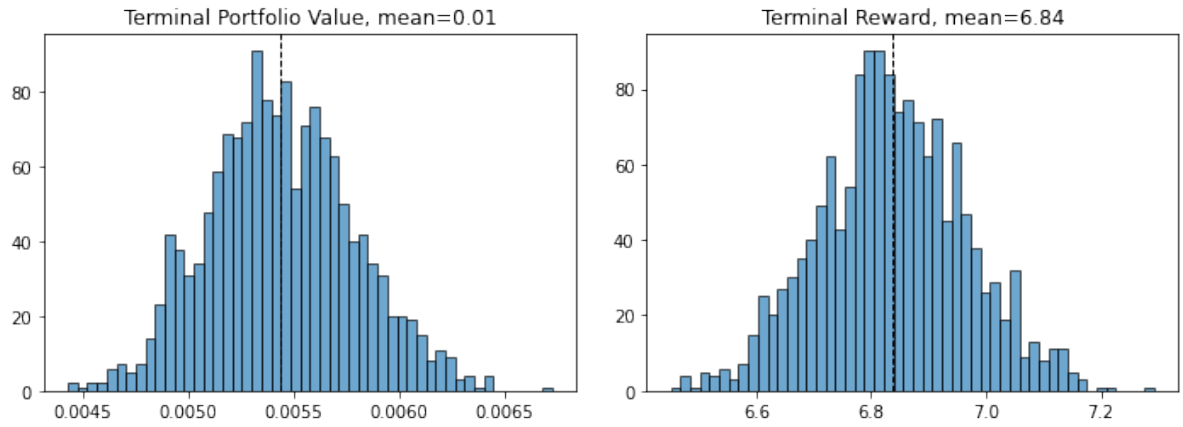


Figure 4.2. Terminal distribution of the portfolio value and the reward for our optimal strategy.

4.2.2 Constant Strategy

$$\pi(t,x) = 1 \tag{4.3}$$

$$c(t,x) = 0 \tag{4.4}$$

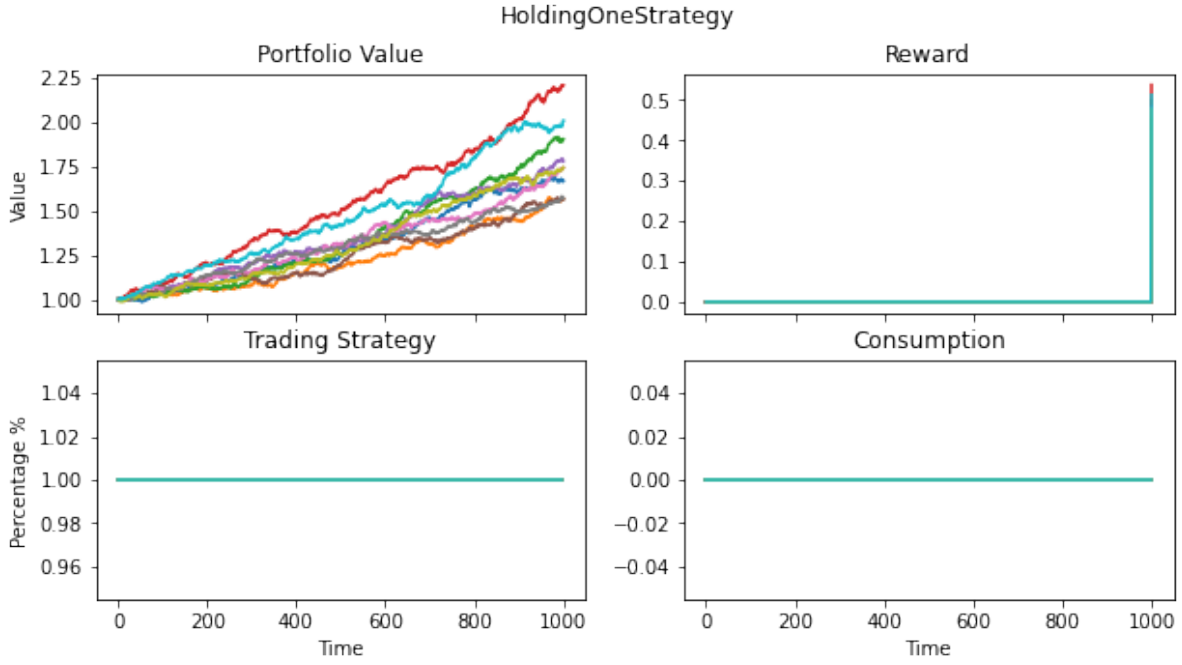


Figure 4.3. Evolution of the portfolio value and the reward for a constant strategy through time, for 10 sampled paths.

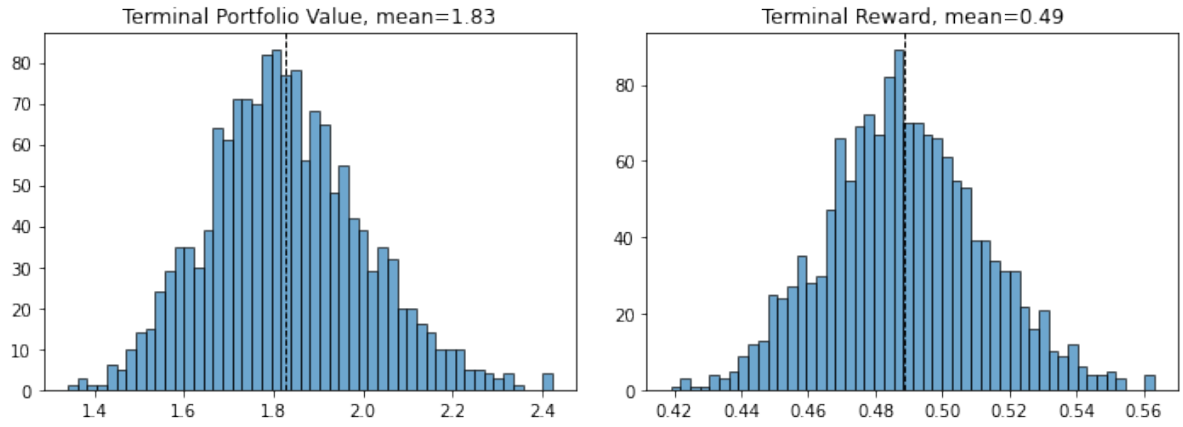


Figure 4.4. Terminal distribution of the portfolio value and the reward for a constant strategy.

4.2.3 Modified Optimal Strategy 1

We make slight modifications to our optimal π.

$$\pi(t,x) = \frac{\sin(t)}{10} + \frac{\mu - r}{\sigma^2 \gamma} \tag{4.5}$$

$$c(t,x) = \begin{cases} \frac{x}{K^{\frac{1}{\gamma}} + T - t} & c = 0 \\ \frac{cx}{(cK^{\frac{1}{\gamma}} - 1)e^{c(t-T)} + 1} & o.w. \end{cases} \tag{4.6}$$

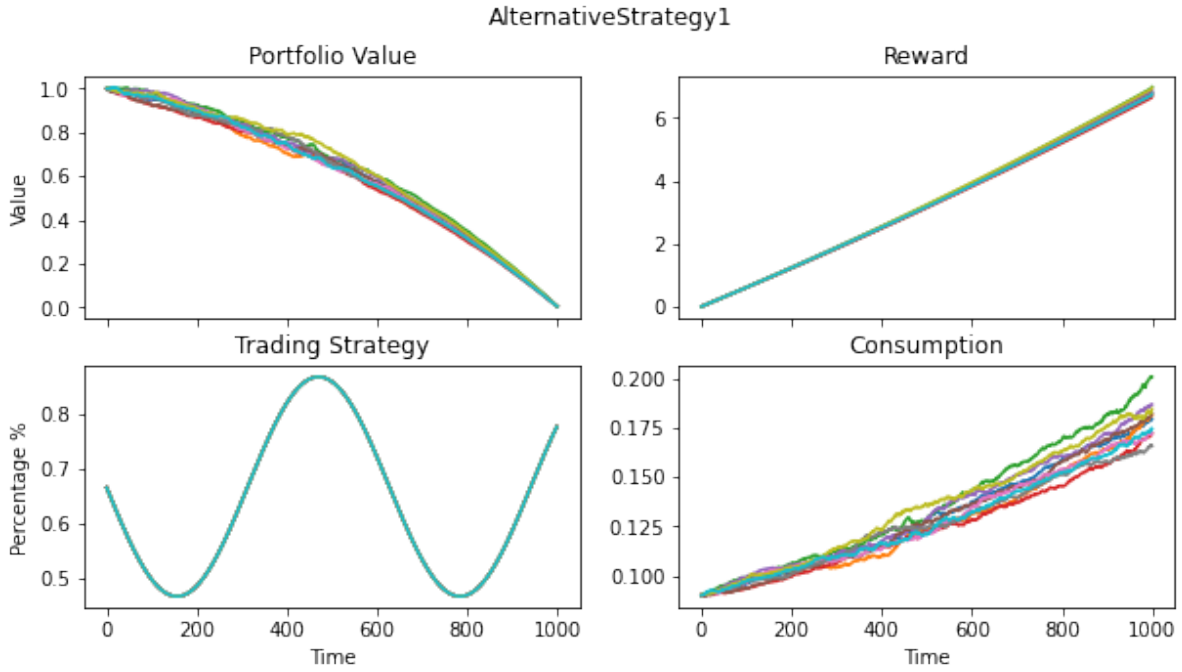


Figure 4.5. Evolution of the portfolio value and the reward for the first modified strategy through time, for 10 sampled paths.

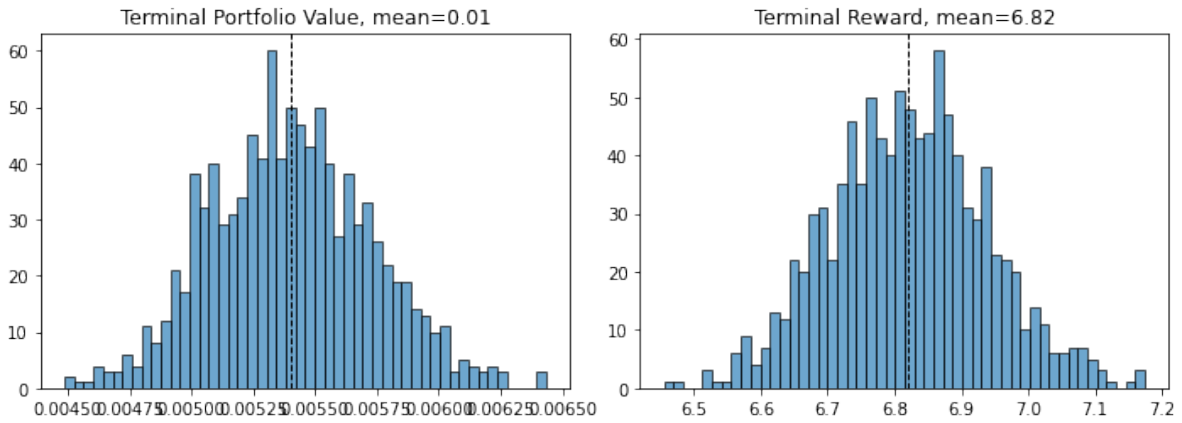


Figure 4.6. Terminal distribution of the portfolio value and the reward for the first modified strategy.

4.2.4 Modified Optimal Strategy 2

We now make a slight modifications to our optimal c.

$$\pi(t,x) = \frac{\mu - r}{\sigma^2 \gamma} \tag{4.7}$$

$$c(t,x) = xe^{c(T-t)} \tag{4.8}$$

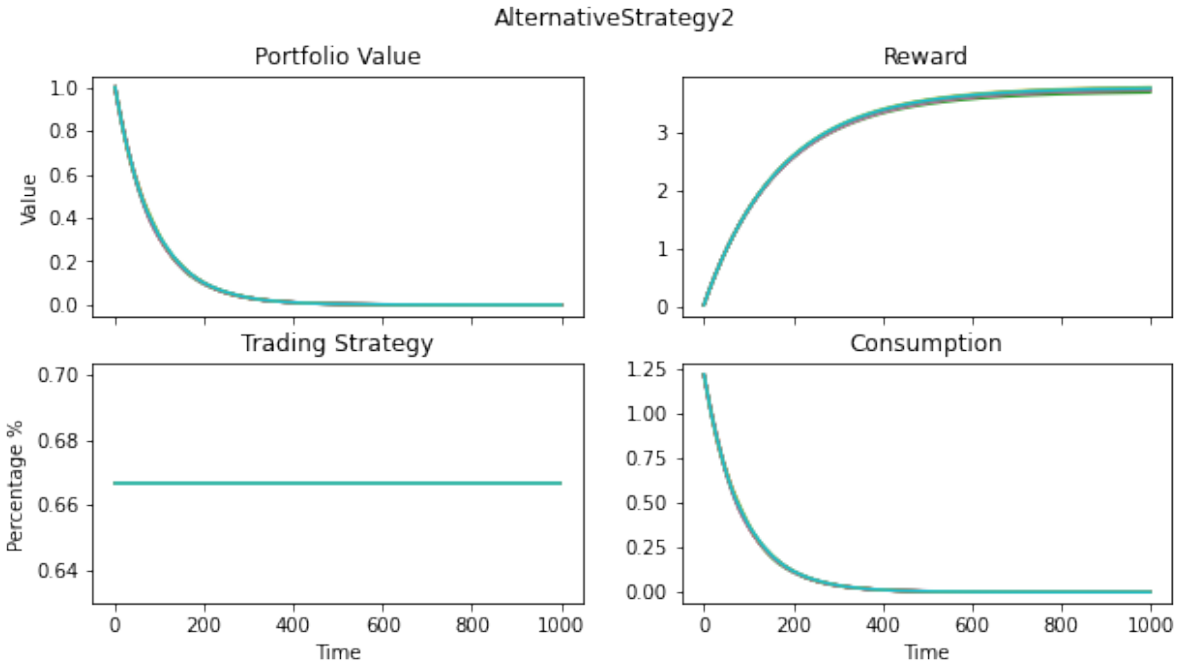


Figure 4.7. Evolution of the portfolio value and the reward for the second modified strategy through time, for 10 sampled paths.

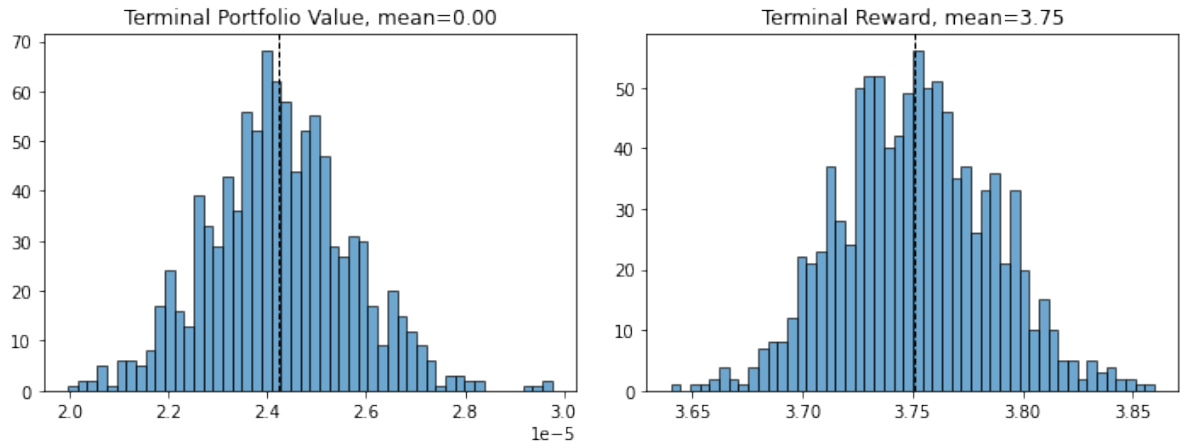


Figure 4.8. Terminal distribution of the portfolio value and the reward for the second modified strategy.

4.2.5 Summary

Strategy	Expected Utility
Optimal Strategy	6.84
Constant Strategy	0.49
Modified 1	6.82
Modified 2	3.75

Table 4.2. Results of the simulation of various strategies. This has been done by simulation of 1000 paths.

As you can see from the table (4.2), the optimal strategy does seem to be optimal as the alternative strategies that we have tried have lower expected utility. However, this does not show that it **is** the optimal strategy. But a small perturbation to the strategy does indeed decrease the expected reward as seen in the first modified strategy.

4.3 Terminal distribution of some quantities studied via simulation

For this specific topic, we will focus on the analysis over hyper-parameters K and γ.

To visualise the dependence of both the expected reward and the expected terminal portfolio value we carried out simulations of 100 paths for 2500 pairs of values of $K \in [0,1]$ and $\gamma \in (0,2]$ (making sure $\gamma \neq 1$)... took more than 4 hours of computation time! We have then plotted the surfaces below.

4.3.1 Terminal Balance X_T

With all the other hyper-parameters fixed, the expected terminal portfolio $\mathbb{E}[X_T]$ increases as K increases. This is because K measures how much the agent cares about the terminal portfolio balance. The more they care about X_T, the more the agent will try to keep a higher terminal balance.

One interesting thing to notice here is that as γ decreases, the agent becomes less risk-averse, the $\mathbb{E}[X_T]$ at terminal time T increase faster as K increase. This does not counter our intuition. Recall that the underlying asset has a non-zero constant excess return $|\mu - r|$ over time t, which means in the extreme case when the agent is risk-seeking $\gamma \to 0$ and $K \to \infty$, an agent can obtain $\mathbb{E}[X_T] \to \infty$ if they want, by either short-selling ($\mu - r < 0$) or borrowing ($\mu - r > 0$) a huge amount, and hence obtaining a large utility at the end.

However, the above explanation seems to encounter a difficulty when $\mathbb{E}[X_T] \to \infty$ only requires $K \to 1$ instead of ∞. One plausible explanation is that the utility functions u_1 and u_2 (1.5) would be identical, when $K = 1$. Note that by eq. (1.4), any utility from consumption is short-term stimulus scaled by dt, where as the long-term reward from X_T is multiplied by 1. In the case when $K \to 1$ and $\gamma \to 0$, it is acceptable that any utility comes from consumption would be strictly worse off if it was deposited into bank account and carried to the terminal state.

This can be roughly demonstrated as follows. First, we have

$$Y_T := \int_0^T e^{rt} c_t \, dt; \quad \text{(If no consumption)} \tag{4.9}$$

$$u_1(x) - u_2(x) \to 0; \quad (\gamma \to 0, K \to 1) \tag{4.10}$$

$$u_1(x+y) = \frac{(x+y)^{1-\gamma}}{1-\gamma} \to x + y; \quad (\gamma \to 0) \tag{4.11}$$

Hence, given that $r = 0.03$ and $\rho = 0.01$ (i.e., $r > \rho$),

$$\int_0^T e^{-\rho t} u_1(c_t) \, dt + e^{-\rho T} u_2(X_T) \tag{4.12}$$

$$< e^{(r-\rho)T} \int_0^T u_1(c_t) \, dt + e^{-\rho T} u_2(X_T); \quad (e^{(r-\rho)T} > 1 > e^{-\rho t}, \forall\, t \geq 0) \tag{4.13}$$

$$< e^{-\rho T} \int_0^T e^{rt} u_1(c_t) \, dt + e^{-\rho T} u_2(X_T) \tag{4.14}$$

$$= e^{-\rho T} Y_T + e^{-\rho T} u_2(X_T); \quad (4.9) \tag{4.15}$$

$$= e^{-\rho T} u_2(X_T + Y_T); \quad (4.10 \text{ and } 4.11) \tag{4.16}$$

we can conclude that the utility comes from consumption indeed even worse off than simply saving everything into the bank account.

In particular, if we plug eq. (2.20) and eq. (2.17) into eq. (1.2), we have

$$dX_t = (\alpha X_t - \frac{X_t}{g_t})dt + \beta X_t dB_t \tag{4.17}$$

$$\alpha := r + (\mu - r)\pi^*, \quad \beta := \sigma\pi^*; \quad \text{(constant)} \tag{4.18}$$

$$h_t := \frac{c^*}{X_t}; \quad \text{(deterministic)} \tag{4.19}$$

Then, substitute $Y_t := e^{\int_0^t \frac{1}{g_u} du} X_t$ into eq. (4.17), we would obtain

$$d\ln(Y_t) = \alpha dt + \beta dB_t \tag{4.20}$$

i.e., $\mathbb{E}[Y_T] = Y_0 e^{\alpha T}$, and hence we have

$$\mathbb{E}[X_T] = \mathbb{E}\left[Y_T e^{\int_0^T -\frac{1}{g_u} du}\right] = \mathbb{E}[Y_T]\, e^{\int_0^T -\frac{1}{g_u} du} = Y_0 e^{\alpha T - \int_0^T \frac{1}{g_u} du} = X_0 e^{\alpha T - \int_0^T \frac{1}{g_u} du} \tag{4.21}$$

where

$$\int_0^T \frac{1}{g_u} du = \begin{cases} \frac{1}{2} T(2K^{\frac{1}{\gamma}} + T) & C = 0 \\ \frac{e^{-CT}(1 - CK^{\frac{1}{\gamma}} + e^{CT}(-1 + C(K^{\frac{1}{\gamma}} + T)))}{C^2} & o.w. \end{cases} \tag{4.22}$$

with C (2.51) be a constant related only to γ. This ensures the previous intuitive explanation from fig. 4.9.

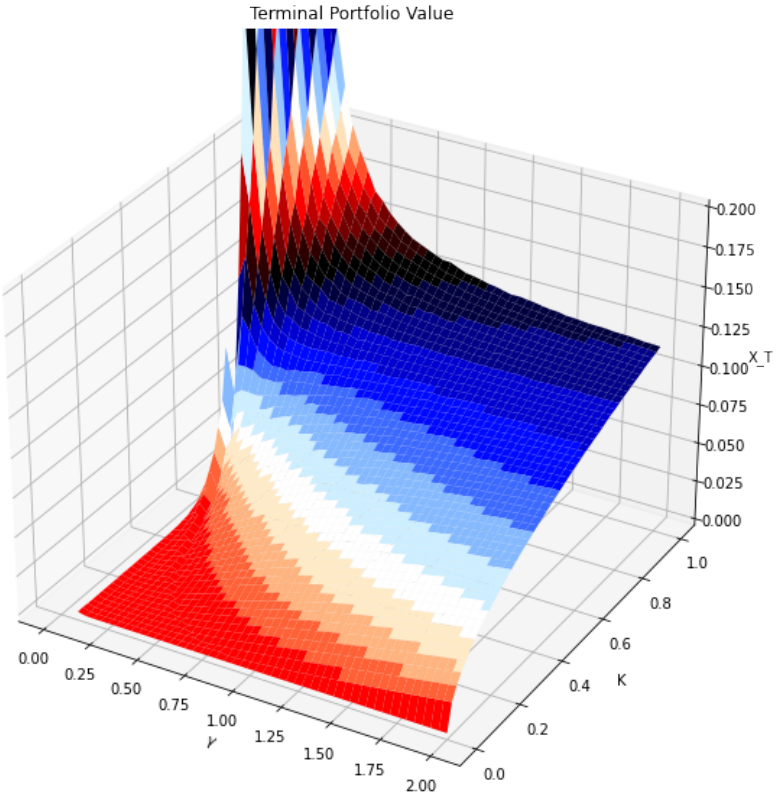


Figure 4.9. Terminal value of expected portfolio value for different values of K and γ.

The obvious feature of the discontinuity when $\gamma = 1$ is clearly seen from the form of the optimal value function (2.52). This behaviour is expected even before solving the problem as the functions u_1 and u_2, that makeup the reward function both have the same behaviour, hence enforcing $\gamma \neq 1$.

A similar analysis to section 4.3.1 in other aspects should be able to be applied to the terminal reward as well. An extra critical point to note here is that the comparison of terminal rewards between different hyper-parameters would not be very informative in economic understanding. This is because the exact values are only valid when comparing with different strategies under the **same** environment (i.e., same hyper-parameters). For example, we can manually set holding a car providing utility 100 v.s. 10k dollars providing utility 10, as well as -100 v.s. -110. But we will not compare that holding 10k dollars is better off in the first scenario than holding a car in the second simply because 10 > -100.

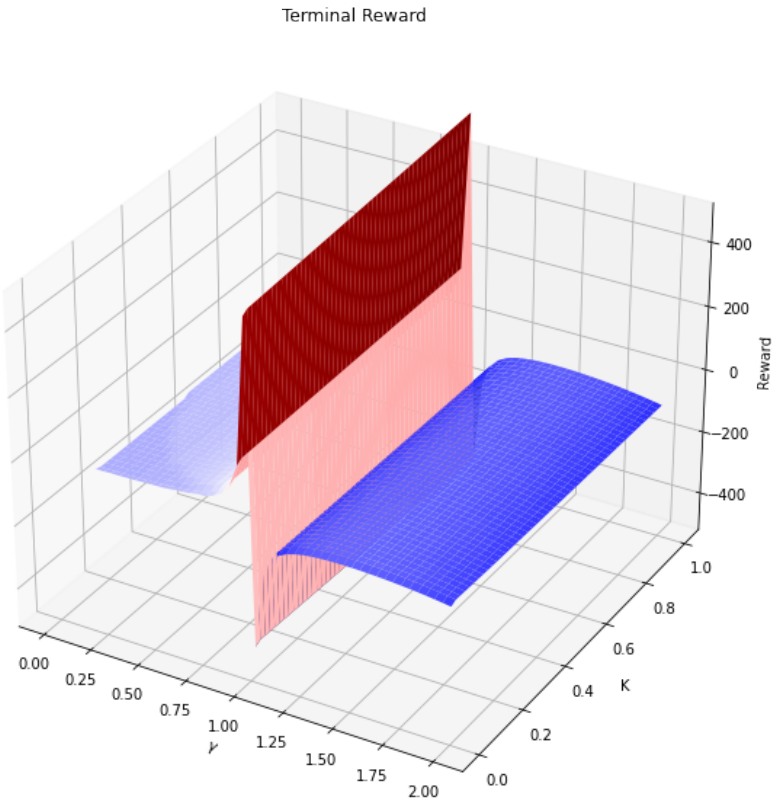


Figure 4.10. Terminal value of expected reward for different values of K and γ. Note: discontinuity for $\gamma = 1$

5 Possible Follow-Up Questions

5.1 Optimal Strategy by Reinforcement Learning

We were able to solve this problem analytically, but many others are much more complicated, or impossible, to solve even with with slight changes to the dynamics or the goal function. Meanwhile, in reality, the dynamics of the environment is not known ahead of time. Therefore, quoted from a wise man in the 21 century, "there does not exist a unique solution to a problem", that motivates us to code a simple reinforcement learning (RL) scheme to find a numerically optimal strategy for this problem and then to compare the performance with the analytically optimal strategy to evaluate the possibility of using reinforcement learning.

To adapt the dynamics for the RL setting, we make a few attempts and adjustments on the rewards scheme. Here, we only presents the final choice we made on the simulation. Further discussion can be found in section 5.1.2.

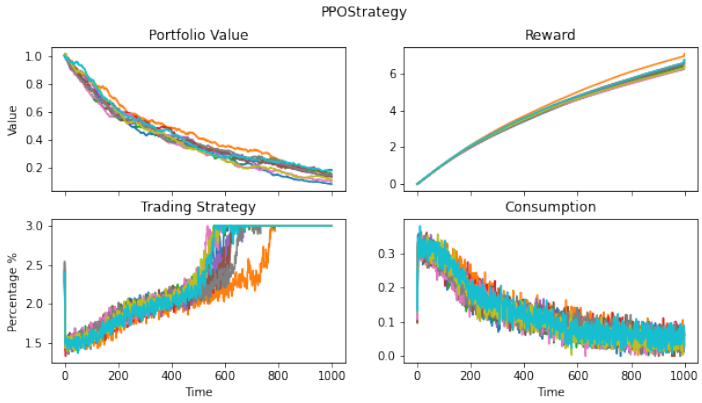
First, to be consistent with the utility function u_1 and u_2 (eq. (1.5)), we force the game to end when the balance account $X_t \leq 0$. Then, the agent will end up with whatever he/she should be rewarded with so far ($\sum e^{-\rho t} u_2(c_t)\Delta t$ and $Ke^{-\rho T} u_2(0)$), but with a penalty for bankruptcy. In the hyper-parameters of this coursework, we hard-coded the penalty to be -10 for simplicity. Further discussion can be found in section 5.1.2. This specific adjustment will not influence our original stochastic control simulation as the optimal strategy would not hit the boundary case when $X_t \leq 0$.

Secondly, we make the consumption as a short-term reward whereas the balance a long-term reward, i.e. the agent would obtain an immediate reward of $e^{-\rho t} u_2(c_t)\Delta t$ after consuming $u_2(c_t)$, but only receive a reward on a balance account X_T when the game terminates. This adjustment also fits into the simulation of the optimal strategies, and this is why there is a small jump at the terminal state T on the rewards graph. The agents task now becomes to find a strategy so that they avoid getting addicted to the short-term consumption, but simultaneously avoid simply waiting and leaving all their assets to the next generation, which is one way to interpret $K = 0.2$, the care of the agent towards their children.

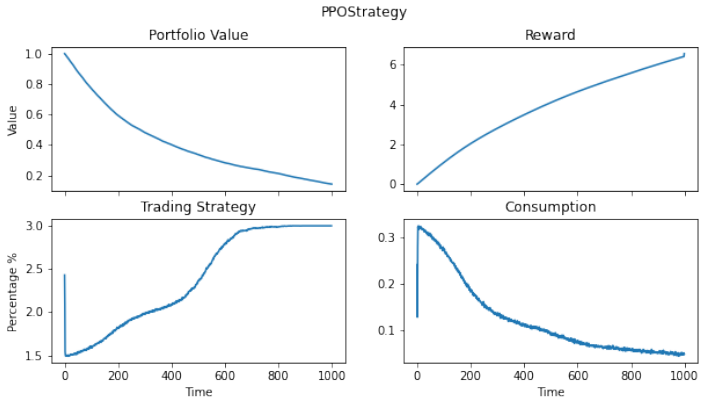
Thirdly, we restrict the action space $\pi_t \in [-3,3]$ and $c_t \in [0,3]$ for the agent to search. This is because we know the optimal strategy is within this range and therefore, as there are fewer possible strategies to try, it makes it easier for the agent to learn. In theory, we can keep the entire action space $\mathbb{R} \times \mathbb{R}_+$ and we should expect a similar convergence result. However, that would involve a huge search space for the agent to exploit, and significantly slow down the convergence speed. To adapt to my poor little laptop surface 6 with only 8 cores and no GPU, it should be acceptable to simplify the search space for my agent.

We have tried several algorithms from DQN (deep Q-learning), PPO (Proximal Policy Optimisation) and Actor Critic, by utilising the open Python-lib Stable Baseline3 [5]. After several comparisons of the convergence speeds and performance, we finally decided to use PPO on this specific environment given the hyper-parameters in table 4.1. A detailed discussion on why we chose PPO and what we found and thought on other algorithms can be found in section 5.1.1.

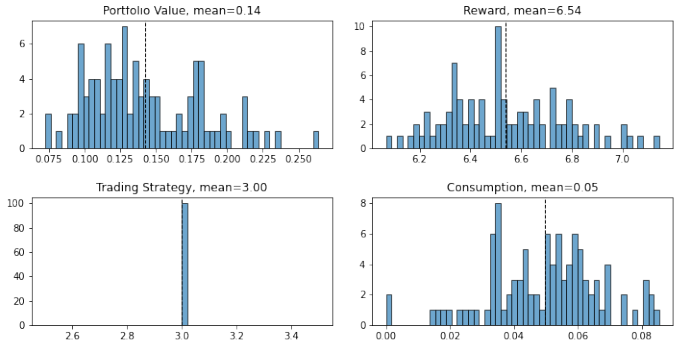
We trained our RL agent over $M \times T = 24000 \times 1000$ time-steps, where $T = 1000$ is the total number of time-steps for one specific game to end, if the agent does not go bankrupt during the game. The agent performs best when $m = 14800$. The performance can be found in fig. 5.1.



(a) Evolution of the portfolio value and the reward, for 10 sampled paths.



(b) Mean of evolution of the portfolio value and the reward, over $n = 100$ simulation.



(c) Terminal distribution of the portfolio value and the reward.

Figure 5.1. The PPO strategy trained with $m = 14800$ time-steps.

One need to note that if we recall the Bellman equation, we quickly realise that the whole point of PPO (and most of other deep reinforcement learning) is trying to estimate the dynamics of the environment, i.e., the probability $\mathbb{P}(r, s'|s, a)$ of received reward r and arrive future state s' conditioning on the current state s and action a. This game here is unfair for the RL agent and the optimal agent because the optimal agent is told what the dynamics are, where as the RL agent is treating the environment as a black box. Therefore, a terminal reward of 6.54 v.s. 6.84 should be viewed as a great result.

However, one thing to notice is that the agent's behaviour is not as we expect in fig. 4.1, even though the agent has achieved a relatively high terminal reward around 6.54 with a reasonable variance compared with the optimal strategy we computed in section 4.2.1. The agent is addicted to the short-term consumption at the beginning, and becomes a wealth hoarder at the end.

One explanation of this is that the RL agent is not able to determine what causes their bankruptcy at the beginning when exploiting the state space randomly. Since they consume randomly at the beginning and may go bankrupt quickly, they tend not dare to consume heavily as approach the end of the game. Instead, they consume massively when they is young, and almost not consume when it becomes old... much like people in real life consuming massively when they are earning a good income but are far from retirement and then having to reduce their spending once they are close to retirement and have to think about the future when they will stop receiving this income. So they must save in order to be able to continue to consume modestly until their passing.

As far as we are aware, such a problem is generally faced by RL game problems. A good analogy of this is the fable about an elephant restrained only by a tiny rope by the trainer, which can be found here.

One of the famous RL papers, by OpenAI on Dota2 [6] in appendix O.2 discussed one potential method to solve this problem results from the long time horizons. In their game, the agent in Dota2, at the early stage, cannot defeat Roshan, a powerful neutral creature in the game. This phenomenon internalised the lesson for the agent never to approach this creature again, even when they had become stronger. Therefore, researchers randomised Roshan's health between zero and the full value, making it easier (sometimes much easier) to kill, so that the agents were still encouraged to challenge Roshan when they can, and also enlarge the search space the agents can potentially reach.

However, one should note that the Roshan in Dota2 is not the same as the consumption in our game. The previous problem is caused by the complexity of the environment, whereas the latter results from the randomness in the environment, especially the true environment defining that the future price of the underlying is indeed following a random walk. The agent might really feel helpless when it is learning given only reward and penalty. After all, shall we expect that a child learns anything given its parents rewarding and penalising it with purely random events happening? If this is the helplessness faced by our agent, I can foresee the desperation faced by me in the future when I become a real-world trader.

Inspired by their paper [6], one notable trail for our project is to randomise the balance account of our agent, i.e., instead of randomising Roshan's health bar, we might try to

randomise the agent's 'health bar'. Once the agent realises how satisfactory it is to consume when they are old (that is, correctly estimates the value of consumption c_t when t is large), they might realise the importance of saving money for the future. But notice that this might collapse the relation between balance account X_t and consumption c_t we wish the agent to learn, which will become a trade-off the researchers faced when making this adjustment. Due to the time limitation for this project, I will leave this work for the future when I become more advanced in reinforcement and machine learning.

In conclusion, this simple routine produced a close to optimal strategy that will surely be improved by more simulations and/or a more sophisticated machine learning algorithms.

5.1.1 Algorithms

Stable Baseline 3 [5] provides a series of different RL algorithms as a API class to train once the environment is well set up. Given our beginner level in deep reinforcement learning, we first pre-trained every RL agent with the algorithms provided by the Stable Baseline 3.

Since our action space is not discrete, the algorithms that fitted our environment and got pre-trained are PPO (Proximal Policy Optimisation), DQN (Deep Q Network), DDPG (Deep Deterministic Policy Gradient), TD3 (Twin Delayed DDPG), and A2C (Asynchronous Advantage Actor Critic). The performance of each algorithm can be found under folder '*fig*' in the zip files. But in short, PPO is the algorithm that possesses a relatively faster learning speed (training time per 1000 time-steps) as well as a promising learning improvement (average rewards increase per 100×1000 games).

Most of the other algorithms either stick to a certain low average reward level (~ 3.5) or stick to the boundary action space ($\pi_t \equiv -3$ and $c_t \equiv 3$). Only the DDPG successfully "found" a better strategy over-performed the optimal stochastic control strategy when the environment allows negative X_t by changing the utility function to

$$\text{sign}(X_T)\frac{|X_T|^{1-\gamma}}{1-\gamma}.$$

This over-performance is due to the fact that one of the technical requirements (i.e., $X_t > 0, \forall\, t$) is violated, hence the optimal stochastic control strategy is not a sub-optimal anymore when $X_t \leq 0$. However, we suspect whether this is found accidentally by DDPG simply because it gets stuck to the boundary of action space again, instead of the algorithm itself.

Therefore, we eventually decided to invest our time into the PPO based on the observed performance in different environment setting. Note that this is the algorithm also applied by OpenAI in the Dota2 experiment [6]. The main reason, which also supported what we found during the pre-training, is that PPO by design is to increase the stability of the RL algorithm by using clipping to avoid too large update. A further description about PPO can be found here.

Although stability and convergence are emphasised by the design of PPO, interestingly, after we started the full training, one can found that the PPO agent during the training tend

to forget the optimal strategy, and started to re-learning everything from the beginning. This will be further discussed in section 5.1.2.

5.1.2 Rewards Scheme and Catastrophic Forgetting

The reward scheme here is simply split into the short term reward from consumption, c_t discounted at each time-step t, and the long term reward from the terminal balance, X_T discounted at terminal time T, purely based on the utility/objective functions u_1 and u_2. However, one needs to note that the reward scheme is essentially a guide for the agent to achieve the target we set up at the beginning. It is designed as an engineering problem that can be polished to potentially improve the performance of the agent. Although it is out of the scope for this assignment, we think it is still worthwhile to discuss.

One shortcoming of current design is that this short-long term rewards setting may confused the agent and increased the difficulty of the training because we are requiring the machine to learn and realise the importance of patience by itself. Refer to the comparison of fig. 4.1 and fig. 5.1, we may claim that our agent failed to learn to be patient.

This failure could result from the simplicity of the build-in neural network in the PPO API of the Stable Baseline 3 (2 layers of 64). Apparently, to improve the complexity of the networks, for instance adapting LSTM to the networks to improve the memory ability of the agents (though the underlying dynamics 1.2 is a Markov process, it could be helpful to memorise their own history in a meaningful way to realise the importance of patience), is a straightforward improvement. However, given the poor condition of our 8 core, 8GB, 0 GPU Surface 6 and the already high cost of training, we could not really take this into consideration.

This motivates us to consider reshaping the reward scheme. Instead of requiring the agent to learn the importance of patience purely from the environment, we can redesign the long term reward $u_2(X_T)$ as

$$e^{-\rho T}u_2(X_T) = \alpha_0 u_2(X_0) + \sum_i [\alpha_{i+1}u_2(X_{t_{i+1}}) - \alpha_i u_2(X_{t_i})]$$

where α_i is the discounting factor we can manually choose to adjust the importance of the utility change from balance X_{t_i} to balance $X_{t_{i+1}}$.

Previously, we have set $\alpha_i \equiv 0$ and $\alpha_T = e^{-\rho T}$. A straightforward trail could be $\alpha_i = e^{-\rho t_i}$ to still keep the consistence of the original objective function. Note that such a reshaping is applying a discounting more as time t increases.

But in general, a big thumb rule would be setting α_i as $\frac{t_i}{T}e^{-\rho T}$, so that the discounting effect would decrease as time t increase. This counter-intuitive design will in fact teach the agent the importance of patience, as the impact is increase as time t increase.

Interestingly so far, one can pre-trained the agent on a general environment quickly that those shaping designs would not significantly improve the performance.

Therefore, given that we encountered the difficulty of balancing consumption and balance already, instead of pushing the long term reward X_T away by discounting it as the objective

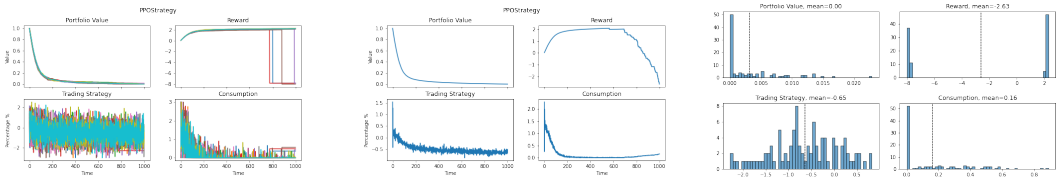
functions u_1 and u_2 indicating, we suggest to follow the intuition of the big thumb rule, but to raise the increment for each time-steps t_i. That is, instead of linearly increase the effect of reward, a plausible design, for instance, could be

$$\alpha_i = (\frac{t_i}{T})^p, \tag{5.1}$$

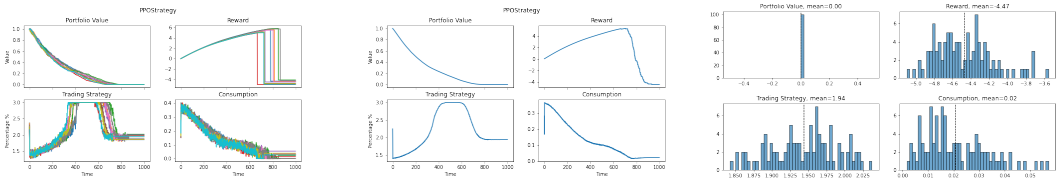
which would at least generate a nearly flat consumption process c_t (instead of downward curved in fig. 5.1) as $p = 2$.

Unfortunately, since the RL algorithm is not the main scope of this assignment, and because we would prefer to have a jump at the terminal state T in the reward process plot (to distinguish $\int u_1(c_t)dt$ and $u_2(X_T)$), we would be happy to leave this discussion as it is right now. The relative code can be found (although not well polished due to time limitation) in the last section in the code marked as "archive".

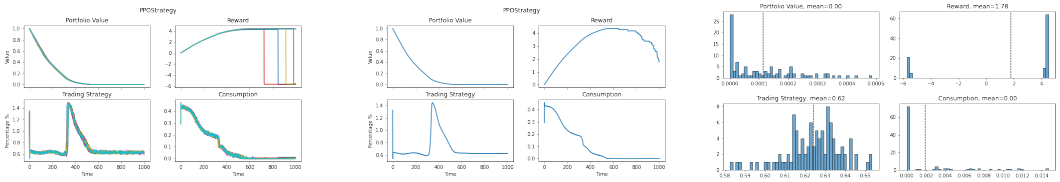
Another thing as we indicated in section 5.1.1, the PPO agent does not manage to keep its memory during the training as we can see in fig. 5.2. Such forgetting can be found more than three times in folder 'figs/PPO-gamma=0.5_K=0.2_rho=0.01_mu=0.06_sigma=0.3_r=0.03' for details.



(a) $m = 100$ time-steps, terminal average reward is -2.63.



(b) $m = 15100$ time-steps, terminal average reward is -4.47.



(c) $m = 21500$ time-steps, terminal average reward is 1.78.

Figure 5.2. The PPO strategy performance during the training process.

Unfortunately, we do not perfectly understand why this PPO agent would experience "Alzheimer's disease" so far. Recall that we have introduced in section 5.1.1 that PPO by design improves the stability and convergence of the training, which makes this discussion meaningful.

One possible hypothesis is to categorise this as a well-known problem in ML called catastrophic forgetting [7]. In short, this happens when the neural networks are optimising the parameters on two, or more than two, different tasks whose interaction of true optimisation parameter sets is empty, so that the training on task 2 will shift the parameters $\boldsymbol{\theta}$ to the optimal parameters sets $\{\boldsymbol{\theta}^2\}$ for task 2, but the non-optimal parameters sets $\{\boldsymbol{\theta}^1\}^c$ for the task 1.

That is, instead of the algorithm itself, it is a problem raised from the neural network behind it. Notice that the task in our assignment is "unique", except that the PPO algorithm is using a window of the total training data set to train the agent. Therefore, suppose this hypothesis is held, it indicates that the optimal parameters are not stationary over time, thus causes the problem. However, it becomes weird that such forgetting only happened when we applied PPO, though it outperforms other algorithms in the maximum historical averaged terminal rewards.

Interestingly, another observation we found is that such "Alzheimer's disease" would disappear when we stop penalising for bankruptcy, i.e., the game would terminal directly and they would not receive a -10 reward for early termination. Note that they would still avoid bankruptcy simply because they should have obtained more reward from consumption if they could stay alive longer. However, as we can see in fig. 5.3, although the agent converges nicely (performs stably over training), it also ends with a mediocre performance (terminal reward converges to around 4). Therefore, we did not present this agent in the previous section. More details about this agent's training process can be found in the folder 'figs/PPO1-gamma=0.5_K=0.2_rho=0.01_mu=0.06_sigma=0.3_r=0.03'.

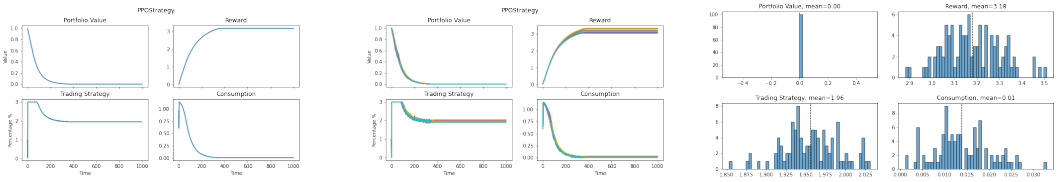


Figure 5.3. The PPO strategy performance given no penalty on bankruptcy, $m = 20000$ time-steps.

In conclusion, reward reshaping can be a nice and in fact important tool to consider for the performance in each specific problem, but might vary from case to case. At our observation, we might consider that there is no such a universal reward shaping design to solve a general RL problem.

5.2 Allowing Negative Consumption

In our work we have assumed, from the start, that the consumption is always positive, representing the fact that we are withdrawing money from the account to cover things such as running costs. However, in the derivation itself, we did not need the assumption. So, we could relax this condition and allow the consumption to be negative, representing the injection of funds to the account. This could be explored but it would not make much

difference to the derivation.

5.3 Allowing Negative Wealth

Throughout our solution, and explicitly in the simulations, we have enforced the agents wealth to be positive. However, if the portfolio value is allowed to take negative values, for $\gamma = 0.5$, the optimal value function will be complex. This same problem will occur for many other values of γ. Therefore providing further motivation for enforcing the positivity of our portfolio value and closing this avenue for investigation without introducing many more restrictions on the values that γ can take.

REFERENCES

[1] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[3] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, and G. Varoquaux, "API design for machine learning software: experiences from the scikit-learn project," in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pp. 108–122, 2013.

[4] W. McKinney *et al.*, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, vol. 445, pp. 51–56, Austin, TX, 2010.

[5] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[6] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with large scale deep reinforcement learning," *CoRR*, vol. abs/1912.06680, 2019.

[7] Wikipedia contributors, "Catastrophic interference — Wikipedia, the free encyclopedia." https://en.wikipedia.org/w/index.php?title=Catastrophic_interference&oldid=1079012547, 2022. [Online; accessed 17-April-2022].